Build your applications with Webpack

alexandrebodin

Published
with GitBook

Table of Contents

Read Me	0
Introduction	1
What is Webpack	1.1
Installation	1.2
Basics	2
Get started	2.1
Configuration	2.2
Using Loaders	2.3
Using Plugins	2.4
Setting up a ReactJS app	2.5
Advanced	3
Setting up an uniserval ReactJS app	3.1
Recipes	4
How to write a loader	4.1
How to write a plugin	4.2
Runing Webpack with Nodejs	4.3
Plugins	5
FAQ	6
Reference	7
Glossary	8
Glossary	

Webpack

This is a work in progress don't hesitate to contribute on the Github repository

This book is an **unofficial documentation** for Webpack with a **learn by example** approach.

Webpack is a great tool I have been using for some time and I love it!

Even though some of the best starter-kits use it, lots of people won't because they find the documentation obscure.

Webpack and it's documentation

Reading the Webpack documentation, I can see why some people think it's not worth it to start using Webpack, but it is!

The learning curve looks steep, but I believe with a more hands on approach, we can level it and make Webpack accessible to all.

Disclaimer

- I am not involved in Webpack's development.
- All thoughts in this book are mine, so don't hesitate to give feedback.
- I feel this tool deserves more love than it receives.
- I believe that if you want better documentation you should contribute somehow.

Prerequisites

I am going to try to start from the beginning and not to leave anyone behind. If you feel some part of this book is difficult to understand just let me know!

- You need to have NodeJS installed.
- I will assume you know basic Javascript.
- I will use `ES2015` syntax for my examples so go and take a look at Babel learn ES2015.

Links

- NodeJs

- npm
- Webpack's official documentation
- The get started tutorial nobody sees
- A nice article by Maxime Fabre

Documentation

- Introduction

Feedback and Contribution

I will be happy to get some feedback and suggestions, If you want to, just open an issue on this book github repository

Thanks

- To Webpack for being an awesome tool and letting me create apps without thinking too much on my development pipeline.

Introduction

- What is webpack
- Installation

What is Webpack

Webpack is a module bundler for the Javascript ecosystem. It's main goal is to Bundle all your application code and dependencies targeting the browser.

What does it do for real ?

Let's say your are using a library like Boostrap and you need to use it in your application. The naive approach is:

```
<!DOCTYPE html>
<html>
  <head>
  <link href="css/bootstrap.min.css" rel="stylesheet">
  </head>
  <body>
    <!-- You would need to load jQuery first because bootstrap requires it  -->
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/1.11.3/jquery.min.js"></scr
    <script src="js/bootstrap.min.js"></script>
  </body>
</html>
```

When your app starts to grow you can see what a hassle it would be to handle all your dependencies.

Now, imagine being able to `require` the Javascript libraries you need directly into your `.js` files like that:

```
var boostrap = require('boostrap');

// ... your code depending on the lib you just loaded
```

And then just have to do that:

```html
<!DOCTYPE html>
  <head>
  </head>
  <body>
    <!-- That's where the webpack magic happens -->
    <script src="js/bundle.js"></script>
  </body>
</html>
```

Well that's what Webpack is all about. It worries about loading and creating a final `js` file to import into your `index.html`

Installation

Webpack

Start by installing Webpack globally

```
npm install -g webpack
```

When developing, it is good practice to add Webpack as a dependency to your project.

```
npm install --save-dev webpack
```

WebpackDevServer

Webpack comes with a package for use while developing.

It allows you to build and serve your files on the fly with auto rebuild when you update your files and some other lovely things!

Go ahead and install it in your project.

```
npm install --save-dev webpack-dev-server
```

Basics

In this section I will try to walk you through the initialization and configuration of a basic Webpack work-flow.

- Get started

Get started

You can find the source code for this example on Github

Installation

First install Webpack globally

```
npm install -g webpack
```

Next create a folder and initialize your npm project

```
mkdir project_dir && cd project_dir
npm init
```

Answer to the questions (if you are not planning on publishing your code you can press enter until it's finished)

Setup

Now let's create some files.

```
project_dir/
-- index.html
-- index.js
-- webpack.config.js
```

- `webpack.config.js` Is the default name for Webpack configuration file. You can setup a full project work-flow just with this file.

In your `index.html` we load the file that Webpack will create for you.

```
<script src="bundle.js"></script>
```

In your `index.js` let's say "Hi".

```
console.log('Hello World!');
```

In your `webpack.config.js` we will initialize the application configuration.

```
module.exports = {
  entry: './index.js',
  output: {
    filename: 'bundle.js'
  }
};
```

The configuration tels Webpack the file it needs to process is `index.js` and it has to output a file `bundle.js` in the `build` directory.

Let's try and run Webpack then !

```
webpack
```

Here Webpack automatically looks for a `webpack.config.js` file. You can specify another file using `'--config=PATH_TO_CONFIG`

Now your project should look like this (ignoring npm stuff)

```
project_dir/
-- bundle.js
-- index.js
-- index.html
-- webpack.config.js
```

In some browsers you cannot load local Javascript files, that's where `webpack-dev-server` comes in.

Among other awesome features `webpack-dev-server` can process your files and create a HTTP server to serve them.

Let's install it in your project

```
npm install --save-dev webpack-dev-server
```

Now run

```
./node_modules/.bin/webpack-dev-server
```

Go to the url `http://localhost:8080` . You should see `Hello World!` in your `console` .

You will notice in your `terminal` that Webpack is still running. If you change your `index.js` file it will reprocess it on the fly. You can just reload your browser then.

Webpack offers features for automatic page reload and much more thanks to plugins and loaders.

Some code

It's all well but for now you have an empty Javascript file. Let's add some code.

Create a folder `app` and a `my_module.js` file in it.

`my_module.js`

```
module.exports = {
  sayHi: function(firstname) {
    console.log('Hi ' + firstname);
  }
};
```

`index.js`

```
var myModule = require('./app/my_module');

myModule.sayHi('Webpack');
```

If you left `webpack-dev-server` running refresh your browser or restart `webpack-dev-server` .

You can see 'Hi Webpack' in the `console` .

To recap, Webpack can load modules just like NodeJs and compile them all into one file. You can now use open-source libraries and organize your code much better.

Npm sugar

Previously you add to use `./node_modules/.bin/webpack-dev-server` instead of `webpack-dev-server` because it isn't installed globally.

Let's setup a `npm` script to simplify this. Add this to the `"scripts"` in your `package.json`

```
{
  // ...
  "scripts": {
    "dev": "webpack-dev-server"
  }
}
```

Now you can just run

```
npm run dev
```

Npm will first look in the `./node_modules` folder of your project when you use modules in your `package.json` scripts so you don't to specify the folder by hand.

Configuration

Using Loaders

Using Plugins

Setting up a ReactJS App

Advanced

Setting up an uniserval ReactJS app

Recipes

How to write a loader

How to write a plugin

Runing Webpack with Nodejs

Plugins

FAQ

Reference

Glossary

Bundle

Basically taking multiple files (your code, JS libraries, and even static resources like images and more...) And package them into one or multiple files that you can just load into your html without ever worring about loading order and duplication.

2.1. Get started 1.1. What is Webpack

Entry

In Webpack the entry configuration allows you to setup the files that will be processed by Webpack.

Loaders

@TODO

2.3. Using Loaders

Plugins

@TODO

5. Plugins 2.4. Using Plugins